

Impact of GenAI and LLMs in Platform Engineering

A Peek Into the Future of Platform Engineering



What We'll Cover?

- Context of AI in Platform Engineering
- Measuring improvements in platform engineering through AI methodologies
- How to improve developer experience through Generative AI
- How is Generative AI improving the observability space?
- What is Agentic AI and how does it impact platform engineering?

How We'll Cover?

- Talk about key concepts
- Look at the tools landscape
- Dive deeper into some tools

Will be switching back-and-forth between concepts and various tools that are coming up in the industry that has made these concepts more real

Context of AI in Platform Engineering

Augment **Platform Engineering with AI** to create a self-healing, adaptive platform that **improves DevEx** autonomously

DIY DevOps → Standardized IDPs → Self-Service → AI-Augmented → Agentic Platforms

Evolution of AI in Platform Engineering

DIY DevOps → Standardized IDPs →

Self-Service → AI-Augmented →

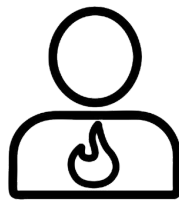
Agentic Platforms

Key Themes in Thoughtworks Tech Radar

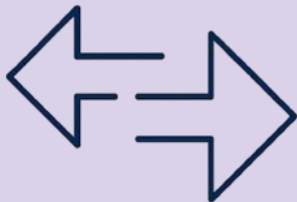


- AI-powered, in-IDE “agentic” coding tools (Cursor, Cline, Windsurf, Claude Code) enable chat-driven prompt-to-code workflows—navigating code, running commands, addressing errors—**under developer oversight**.
Highlight: this is not full automation—human supervision remains essential to ensure quality and prevent complacency.
- Observability is expanding to cover **LLM & AI performance**, with tools like Weights & Biases Weave, Arize Phoenix, Helicone, and HumanLoop.
- RAG remains central for effective GenAI workflows. Variants include **Corrective-RAG**, **Fusion-RAG**, **Self-RAG**, and graph-based retrieval (FastGraphRAG).
- Focus shifts from raw volume to **rich, unstructured data management**—especially for AI-ready systems.

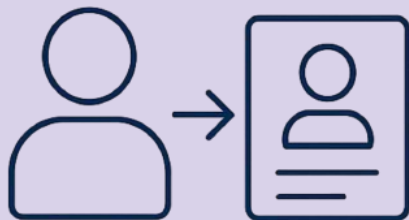
Shift in Developer Landscape



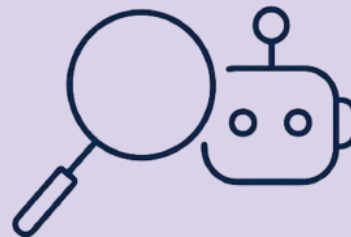
Developer Role
Operator → Supervisor



Reactive → Proactive



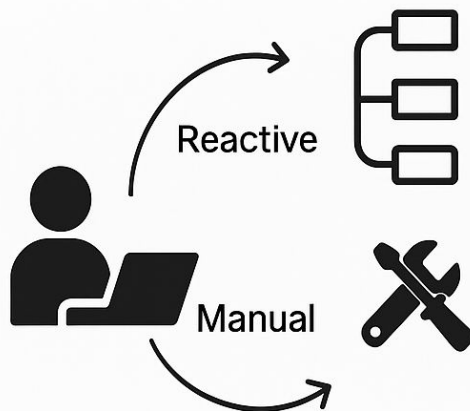
Generic → Personalize



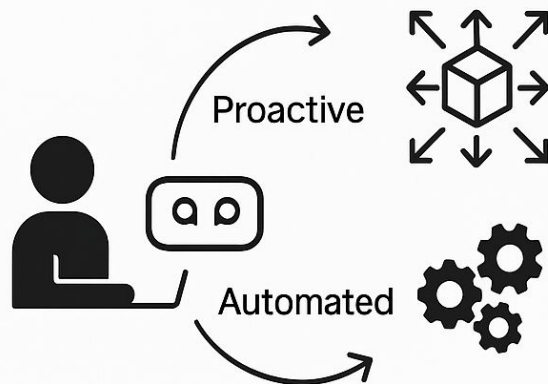
Inferred → Autonomous

How is the evolution happening?

Old Developer Workflow



New AI-Augmented Workflow



“Companies using AI to enhance developer experience report up to a **35% improvement in developer productivity** and a **20–30% reduction in time to deploy software**.” (McKinsey 2023)

“The future of platform engineering is not just self-service—it’s **self-evolving**. AI-powered platforms that learn from developer behavior will define the next wave of enterprise software delivery.” (Forrester 2024)

Scope of AI in Platform Engineering

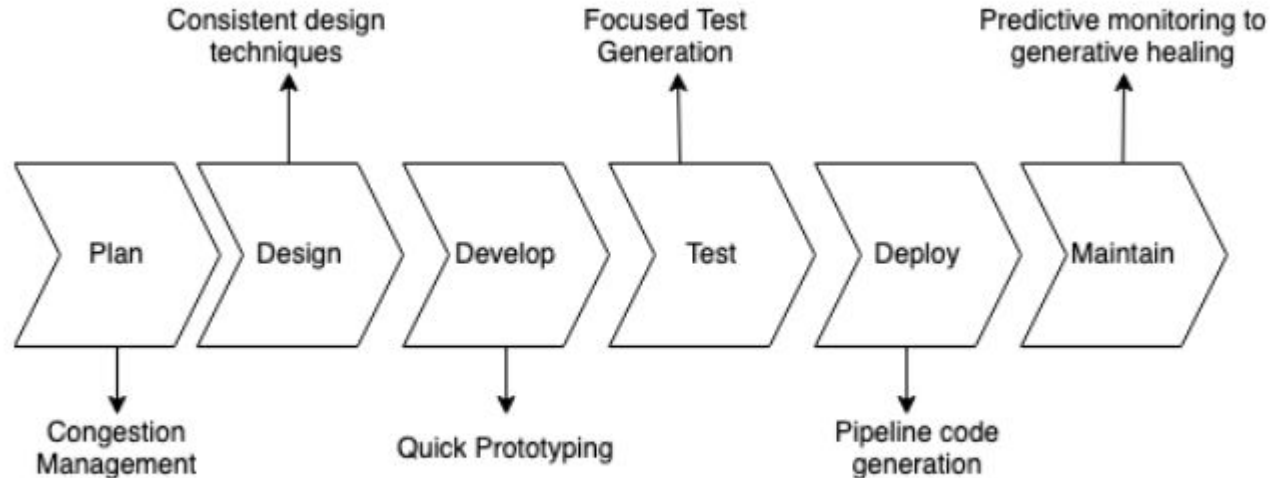
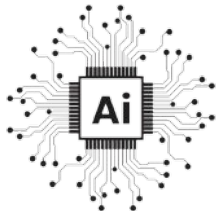


Figure 11.1 Focus Areas for Generative AI in Platform Strategy. This is a rapidly evolving space, and focus areas should be identified using the same product strategies discussed throughout this chapter.

Gen AI (and related) hype



Vibe Coding

AI-driven development where natural language prompts generate code.

Devin

An AI software engineer capable of autonomously completing development tasks.

Cursor

An AI-enhanced code editor integrating LLMs for real-time coding assistance.

Sweep

An AI tool that automates codebase maintenance and bug fixes.

Windsurf

A free AI-powered code completion tool supporting multiple programming languages.

RAG

Technique combining external data retrieval with AI-generated responses for context-aware coding.

AutoDev Agents

Autonomous AI agents managing end-to-end software development processes.

AI Code Review

Automated code review systems utilizing AI to detect issues and suggest improvements.

MCP

A standardized way for AI agents to interact with external data sources and tools

AI CI/CD Pipelines

Integration of AI in Continuous Integration/Continuous Deployment for automated software delivery.

Developer Pain Points

Aspect	Pre-Platform Engineering	Platform Engineering	AI-Augmented Platforms
Tooling	DIY Toolchain Assembly	Standardized Toolchains (IDPs)	Embedded Intelligent Agents (Proactive Helpers)
Ops Model	Ticket-based Ops (manual)	Self-Service Portals (partial)	Contextual, Autonomous Workflows (intent-based)
Feedback	Late, Manual, Error-Prone	Faster CI/CD Feedback, but disconnected	Instant Feedback Loops with Explainability (AI Observability)
Cognitive Load	High: developers manage infra details	Medium: golden paths help reduce burden	Low: Agents reduce toil and complexity dynamically
Toil and Reactive Work	Frequent firefighting, manual debugging	Some automation, alerting systems	Self-healing agents, proactive remediation
Sense of Impact	Poor: disconnection from business outcomes	Medium: metrics dashboards accessible	High: platforms surface business value of dev work (OKRs tied to pipelines)
Developer Experience	Frustrating, disjointed	Improved, but still fragmented	Flow State first, collaborative human+agent experience

Tools Landscape

AINative*Dev*

Home News Podcast About Landscape AINDCon2025

Landscape

Stats
353 tools
+ Submit

Last updated
Friday
4 July

Your Guide to the AI Development Ecosystem

Show all

AIE

Autonomous Agent

Beta

Browser

Code

Code Benchmark

Compliance

Design

DevOps

DevSecOps

Documentation

Editor

Execution Sandbox

Frontend & Mobile

GA

Gateway

Infrastructure As Code

MCP

Migration

Model

Nocode

Observability

OSS

Product

Prompting

Prototyping

Quality Assurance

Requirements

Review

Spec Driven

SRE

Terminal

Testing

Verified

Vuln Scanning

Grid

Catalog

List


Q Search...


Product


Requirements16


Design11


Nocode19


AISPEC


CARROT


CHAT PRD


CLOUD ARCHITECT..


BUILDER


LOCOFY


ANIMA


CHORDIO


ARDOR


CHEF


CREATEMVP


DALTON


CODEGUIDE


CREATEMVP


DELIBR


DOCUOPIA


CLONE UI


CODE


DESIGN2CODE


FIG4AI


DATA BUTTON


FLUTTERFLOW

FRAMER


JO


FULL STACK ROADMAP


LUNA AI


PRODUCT MONKEY


Show more [5]


FIGMA.AI

GALILEO

VISILY

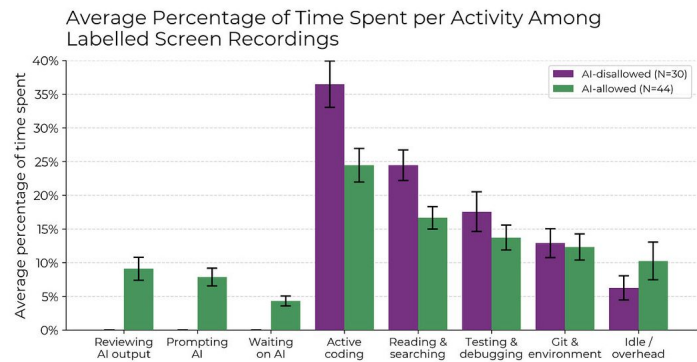
KEAK

LEAP

LEAPTER

Show more [8]

METR Study in July 2025



Using AI meant less time spent coding, but the work took longer, overall. Source: [METR](#)

Developers are over optimistic in their estimates about AI's productivity impact – initially, at least. From the survey:

"Both experts and developers drastically overestimate the usefulness of AI on developer productivity, even after they have spent many hours using the tools. This underscores the importance of conducting field experiments with robust outcome measures, compared to relying solely on expert forecasts or developer surveys."

Factors likely to contribute to slowdown

Factor	Type	Relevant Observations
Over-optimism about AI usefulness (C.1.1)	🕒	<ul style="list-style-type: none">Developers forecast AI will decrease implementation time by 24%Developers post hoc estimate AI decreased implementation time by 20%
High developer familiarity with repositories (C.1.2)	👤	<ul style="list-style-type: none">Developers slowed down more on issues they are more familiar withDevelopers report that their experience makes it difficult for AI to help themDevelopers average 5 years experience and 1,500 commits on repositories
Large and complex repositories (C.1.3)	🏢	<ul style="list-style-type: none">Developers report AI performs worse in large and complex environmentsRepositories average 10 years old with >1,100,000 lines of code
Low AI reliability (C.1.4)	🏢	<ul style="list-style-type: none">Developers accept <44% of AI generationsMajority report making major changes to clean up AI code9% of time spent reviewing/cleaning AI outputs
Implicit repository context (C.1.5)	🏢👤	<ul style="list-style-type: none">Developers report AI doesn't utilize important tacit knowledge or context

**Measuring improvements in platform
engineering through AI methodologies**

Evolving Metrics with AI in Platform Engineering

- Move beyond DORA/SPACE: AI enables *context-aware, continuous measurement*.
- Use AI/ML to correlate productivity signals with operational data (e.g., CI/CD logs + developer sentiment).
- AI-based time-series clustering helps detect anomalies in flow metrics across teams or domains.
- Predictive analytics: Forecast developer bottlenecks before they escalate.



FLOW
ENGINEERING

AI Powered Measurement Techniques

- Embedding feedback loops via AI agents to track developer satisfaction (e.g., nudges, micro-surveys).
- RAG pipelines (Retrieval-Augmented Generation) help summarize platform telemetry and adoption data.
- NLP techniques to analyze developer tickets, platform docs, and Slack threads for friction patterns.
- Use reinforcement learning models to simulate policy impact on performance (e.g., golden path optimization).

AI Driven SDLC

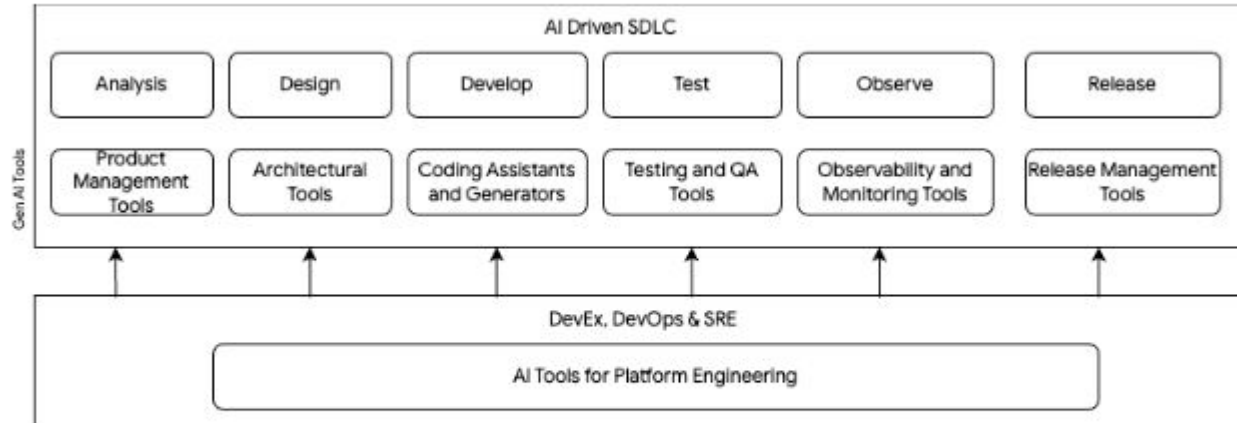


Figure 11.3 A high-level view of AI-driven SDLC that improves Developer Experience through DevOps principles, making it easier to do Site Reliability Engineering by applying platform engineering principles

A Metrics Matrix - DORA

Category	Traditional DevOps Metric	AI-Augmented Signal	How AI Adds Value
Delivery Performance	Lead Time for Changes	Predicted Lead Time (based on PR size, sentiment, workload history)	Predict delays before they happen using time-series forecasting
	Deployment Frequency	Adaptive Deployment Cadence	Adjusted based on inferred team stress, test coverage, and feature risk
Reliability	Change Failure Rate	Contextual CFR by Domain, Developer Persona, Time of Day	Cluster-based breakdowns, pattern detection in test/log failures
	MTTR (Mean Time to Recovery)	RCA Latency, Alert-to-Action Time, MTTR Forecasting	LLM-assisted RCA and root cause suggestions via incident summaries

A Metrics Matrix - Other Leading & Lagging

Category	Traditional DevOps Metric	AI-Augmented Signal	How AI Adds Value
Quality	Bug Count, Test Coverage	Anomaly Detection in Test Flakiness, Predictive Defect Hotspots	AI flags brittle test patterns or bug-prone code via embeddings
Developer Productivity	PR Cycle Time, Commit Volume	Friction Index (based on review churn, rework, sentiment)	NLP on PR comments + ticket threads + velocity metrics
Developer Experience	Survey-based NPS or satisfaction scores	Real-time DevEx Score (via agent prompts, workflow lag, behavioral patterns)	Embedded micro-surveys + telemetry inference using GenAI agents
Operational Efficiency	Ticket Volume / SLA	Task Automation Rate, AI Intervention Effectiveness	% of tasks handled autonomously; learning loops from past interventions
Observability	Alert Count, SLO Violations	Narrative-based Alert Summaries, Failure Pattern Clustering, Predictive Incident Models	Reduce noise, surface actionable signals with GenAI summarization
Security	Vulnerability Scan Results	Contextual Risk Score + Auto-Remediation Suggestions	LLM-assisted interpretation of security findings
Cost Governance	Monthly Cloud Spend Reports	Real-time Cost Efficiency Score by Team/Service	AI-driven recommendations to optimize cost per deploy/unit/test

CoPilot Adoption is suboptimal



The business case for GitHub Copilot

How to measure downstream
impacts?

The AI-Native Engineering
Intelligence Platform



0:03 / 2:06



How to improve developer experience through Generative AI

GenAI for Frictionless DevEx

- GenAI copilots offer **personalized guidance** across onboarding, CI/CD config, and environment setup.
- Inline code explanation and remediation (via IDE-integrated LLMs) reduce cognitive load.
- Natural language interfaces to developer portals and documentation make platforms more discoverable.
- Context-aware recommendations based on team/project history.

Platform Embedding and GenAI Fusion

- Use LLMs + vector embeddings to build semantic search across platform services and infra policies.
- Automate environment provisioning using prompts (e.g., “Create a staging env with Kafka + Redis”).
- Intelligent platform feedback: AI agents suggest faster workflows based on past usage patterns.
- Developer telemetry + LLM summarization = on-demand insights into developer pain points

Example: Pulumi CoPilot

Pulumi Copilot

Intelligent Cloud Management



Example: Robusta RCA

The screenshot displays the Robusta RCA interface, which is used for analyzing system events and alerts. The interface is divided into several sections:

- Header:** Shows the Robusta logo and the current cluster name, "natan-openshift".
- Search and Filter:** Includes a search bar for "Type / to search" and filters for "Cluster", "Namespace", and "App Name".
- Timeline:** A horizontal timeline showing various events over time. The events are categorized by severity (Low, Medium, High) and type (e.g., KubeContainerWaiting, KubeDeploymentRolloutStuck, KubeJobFailed).
- Event Details:** A detailed view of a specific event, "KubeContainerWaiting", which occurred on Oct 28, 2024, at 17:33:45. The event is described as "pod/customer-relations-webapp-5d98ffcd-td5gz in namespace default on container crw-main-container has been in waiting state for longer than 1 hour. Pod is not ready due to ImagePullBackoff ImagePullBackOff: Back-off pulling image 'yourcompany/crw:latest'".
- Event Stream:** A table at the bottom showing a list of events, including their cluster, namespace, event name, and a link to view the event details.

The interface also includes a sidebar with navigation icons and a bottom status bar showing the current time and system information.

How is Generative AI improving the observability space?

AI-Enhanced Observability Patterns

- GenAI summarization reduces alert fatigue by transforming logs/traces into **actionable narratives**.
- Detecting unseen failure patterns via LLM-powered anomaly detection and log clustering.
- Root cause analysis (RCA) through prompt-based investigation: “Why did checkout fail at 3AM?”
- Auto generated incident reports via RAG workflows trained on runbooks + prior incidents.

Impact on Observability Tools



Figure 11.7 A classification of some of the most popular observability tools in the industry today and how they fare against each of the five axes of comparisons we identified.

Closing the Loop with AI in Observability

- Event correlation: LLM agents group related alerts across systems and suggest resolutions.
- LLMs offer **plain-language dashboards**, lowering observability complexity for non-experts.
- Prometheus/Grafana integrations with chat-based GenAI assistants for deeper queries.
- Continuous training from production data to evolve observability agents' accuracy over time.

RAG with prompting

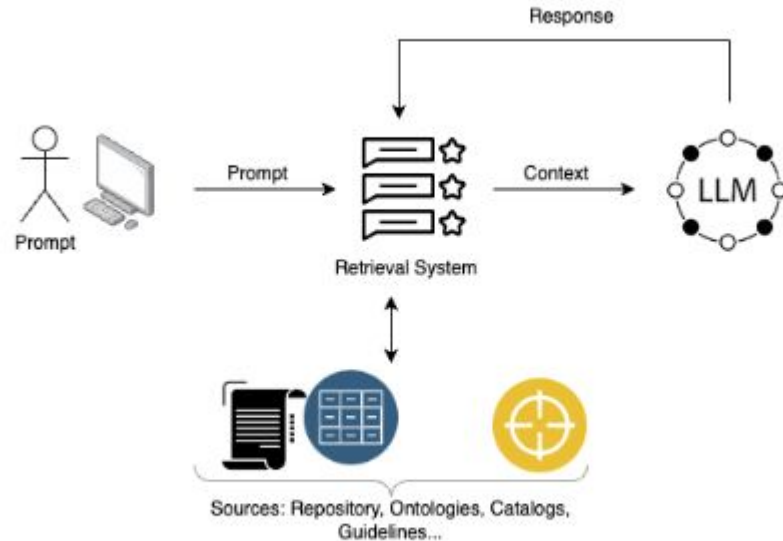


Figure 11.4 A simple view of how Retrieval Augmented Generation works within a prompting process

Automating Software Operations using Resolve.ai

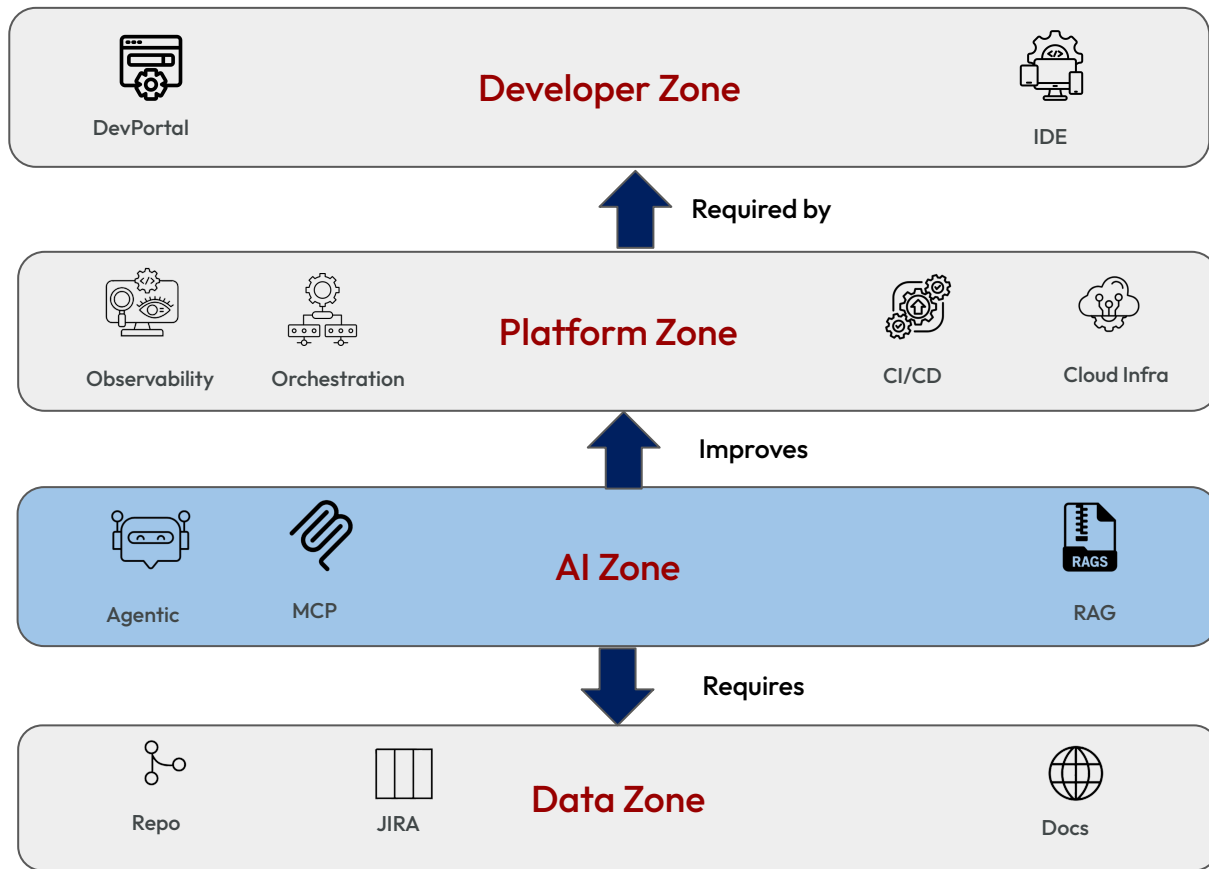


Agents to do what you need in PE Space



**What is Agentic AI and how does it
impact platform engineering?**

A Notional Architecture



Defining Agentic AI in Platform Context

- **Agentic AI** = autonomous systems that perceive, reason, and act with a goal-oriented approach.
- Unlike static GenAI, agentic systems evolve workflows (e.g., tune CI pipelines, suggest infra shifts).
- Examples in platform engineering: infra remediation bots, proactive security hardening agents.
- Builds on RLHF (Reinforcement Learning with Human Feedback) to improve decision quality over time.

Impact on Platform Engineering Operating Model

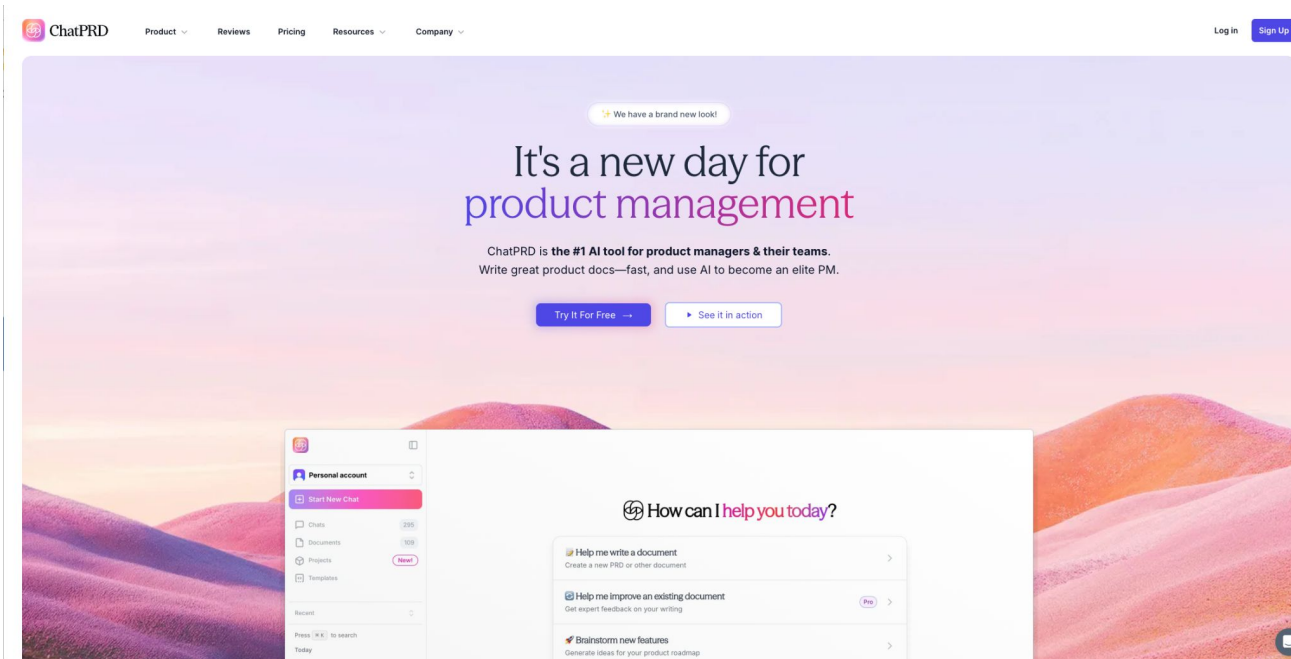
- Platform teams shift from scripting tools to **designing agent ecosystems**.
- Facilitates *intent-based interfaces*: e.g., “Optimize cost for this app while maintaining SLOs.”
- Encourages composability: Agents can interact (DevEx agent ↔ Cost Optimizer agent).
- Supports “Platform as a Brain” vision—autonomous agents that evolve platforms without manual config.

Agentic AI with Engineering Platform

Use Case	With Engineering Platform	With EP + Agentic AI
Fixing Production Issues	<ul style="list-style-type: none">Resources provisioned with platform capabilities manuallyLogs, metrics and incident workflows available from the platform	<ul style="list-style-type: none">Agents proactively detect, diagnose, and resolve issuesFixes are applied, tested and deployed autonomously
Pipeline Optimization	<ul style="list-style-type: none">Standard pipelines reduce variabilityManual interventions are needed for test optimizations and rollbacks	<ul style="list-style-type: none">Dynamically optimize test execution, deployment strategies and auto-trigger rollbacks
Observability & Incident Response	<ul style="list-style-type: none">Correlates logs, metrics and traces automaticallyManual analysis and actions needed	<ul style="list-style-type: none">Predictive anomaly detection prevents incidents before they occurAgents autonomously remediate issues and alerts
Cost & Governance Automation	<ul style="list-style-type: none">Cost reports and compliance policies are generated from pipelinesRemediation requires additional manual/automated approaches	<ul style="list-style-type: none">Continuously optimizes resource allocation in realtimeSecurity policies are autonomously enforced

Agentic AI to help create PRD

- We will use a tool called chatprd.ai for demonstration purposes



Agentic AI to help create PRD

- Step # 1: Initial Prompt

The screenshot displays the ChatPRD web application interface. On the left is a sidebar with a 'Personal account' dropdown, a 'Start New Chat' button, and navigation links for Notifications (0), Chats (1), Documents (0), Projects (New!), and Templates. Below these, it shows 'You have used 1 of 3 free chats.' with a progress bar and an 'Upgrade for \$5 / month' link, followed by an 'Upgrade Now' button. The 'Recent' section lists 'All chats' and a search bar with the text 'Press [K] to search'. A recent chat entry is visible: 'PRD for PR Review Agent Crea...'. The main chat area shows a conversation with the user's prompt: 'Help me create a PRD for an agent that will do a PR Review'. The AI's response begins with 'Absolutely, Ajay! Creating a PRD for an automated agent that conducts PR (pull request) reviews sounds like a strong step toward improving developer productivity and code quality on Platformetrics.' and continues with a list of clarifying questions: 1. **Target Users:** Who is the main audience for this agent? (e.g., internal Platformetrics engineers, open-source contributors, 3rd-party partners, etc.) 2. **Review Scope:** Are you envisioning a purely code-style/linting review, or should the agent give architectural suggestions, catch regressions, enforce security patterns, etc.? 3. **Integration Point:** Which platforms or tools should this work with? (e.g., GitHub, GitLab, Bitbucket, custom git workflows?) 4. **Agent Behavior:** Should the agent auto-approve, just comment, block merges, or escalate issues? 5. **"MVP" vs. Ambitious:** Are you looking to ship a slimmed-down "first version" quickly or expect broader capabilities in the first release? The response concludes with 'If you give me your quick thoughts (even rough!), I'll build a tailored PRD draft that fits your real needs and roadmap.' and 'How do you see your team and product benefiting most from this agent? Any "must-haves" or red lines you want in (or out)?'. At the bottom, there is a 'Send a message' input field and a status bar indicating 'You are using the ChatPRD: PRD template'.

Personal account

Start New Chat

Notifications 0

Chats 1

Documents 0

Projects New!

Templates

You have used 1 of 3 free chats.

Upgrade for \$5 / month

Upgrade Now

Recent All chats

Press [K] to search

Today

PRD for PR Review Agent Crea...

Project / PRD for PR Review Agent Creation 207 of 3000 words used

Share 0

Help me create a PRD for an agent that will do a PR Review

Ajay

Free Trial

Absolutely, Ajay! Creating a PRD for an automated agent that conducts PR (pull request) reviews sounds like a strong step toward improving developer productivity and code quality on Platformetrics.

Before I get started, a couple of clarifying questions will help make the document laser-focused and actionable:

1. **Target Users:** Who is the main audience for this agent? (e.g., internal Platformetrics engineers, open-source contributors, 3rd-party partners, etc.)
2. **Review Scope:** Are you envisioning a purely code-style/linting review, or should the agent give architectural suggestions, catch regressions, enforce security patterns, etc.?
3. **Integration Point:** Which platforms or tools should this work with? (e.g., GitHub, GitLab, Bitbucket, custom git workflows?)
4. **Agent Behavior:** Should the agent auto-approve, just comment, block merges, or escalate issues?
5. **"MVP" vs. Ambitious:** Are you looking to ship a slimmed-down "first version" quickly or expect broader capabilities in the first release?

If you give me your quick thoughts (even rough!), I'll build a tailored PRD draft that fits your real needs and roadmap.

How do you see your team and product benefiting most from this agent? Any "must-haves" or red lines you want in (or out)?

You are using the ChatPRD: PRD template

Send a message

Agentic AI to help create PRD

Step # 2: Refine with specifics

The screenshot shows a chat interface with an AI agent named Ajay. The interface includes a sidebar on the left with navigation options: Personal account, Start New Chat, Notifications (0), Chats (1), Documents (0), Projects (New!), and Templates. A status bar indicates 'You have used 1 of 3 free chats.' and an 'Upgrade Now' button. The main chat area shows a conversation with Ajay, who is a 'Free Trial' user. The chat title is '+ Project / PRD for PR Review Agent Creation' with a note '445 of 3000 words used'. The chat history shows a previous message from Ajay asking for quick thoughts and a follow-up question about team and product specifics. The current message from the user provides details: 'Target Users: Internal developers of my team who are building this agent for potential external users', 'Scope: Provide architectural suggestions, and security challenges', 'Integration: It should work with Github', 'Behavior: Yes, auto-approve but show the flow correctly', and 'Want to create an MVP Version...'. Ajay's response provides a summary of the direction for the MVP PR Review Agent, listing key points: Target users (internal devs), Scope (architectural suggestions & security), Integration (Github PRs), Behavior (auto-approve with transparency), and Release (start lean). He then provides an 'Outline Proposal' with 9 numbered items: 1. TL;DR, 2. Goals, 3. User Stories, 4. Functional Requirements, 5. User Experience, 6. Narrative, 7. Success Metrics, 8. Technical/Integration, and 9. Milestones & Team. The interface also shows a 'Send a message' input field at the bottom and a 'Share' button in the top right.

Project / PRD for PR Review Agent Creation 445 of 3000 words used

Share

Personal account

Start New Chat

Notifications 0

Chats 1

Documents 0

Projects (New!)

Templates

You have used 1 of 3 free chats.

Upgrade for \$5 / month

Upgrade Now

Recent All chats

Press **X** **X** to search

Today

PRD for PR Review Agent Crea...

Ajay Free Trial

If you give me your quick thoughts (even rough!), I'll build a tailored PRD draft that fits your real needs and roadmap.

How do you see your team and product benefiting most from this agent? Any "must-haves" or red lines you want in (or out)?

Target Users: Internal developers of my team who are building this agent for potential external users
Scope: Provide architectural suggestions, and security challenges
Integration: It should work with Github
Behavior: Yes, auto-approve but show the flow correctly
Want to create an MVP Version...

Perfect, thanks for the detail, Ajay! Here's a summary of your direction for the MVP PR Review Agent:

- **Target users:** Your internal devs first (with sights on external customers eventually)
- **Scope:** Focused on architectural suggestions & security challenges (not just simple linting)
- **Integration:** Must work with Github (PRs)
- **Behavior:** Can auto-approve, but transparency of actions/decisions is important
- **Release:** Start lean—minimum viable product

Here's how I suggest we structure the PRD for clarity and momentum. Does this match your expectations before I dive into the draft?

Outline Proposal:

1. **TL;DR:** Short summary of problem/solution
2. **Goals:** Business/user goals, non-goals (for focus)
3. **User Stories:** Developer workflow from both the dev building it and hypothetical external user
4. **Functional Requirements:** Key features—Github integration, code parsing, review feedback, auto-approval logic, transparency/logging
5. **User Experience:** End-to-end Journey from PR creation to review to approval/notification
6. **Narrative:** Example scenario showing user value
7. **Success Metrics:** Adoption, speed, security, etc.
8. **Technical/Integration:** Github API, code analysis layers, etc.
9. **Milestones & Team:** Lean team/schedule/phases


Give me a thumbs up (or suggest tweaks) and I'll draft the full PRD in your template!

You are using the ChatPRD: PRD template

Send a message

Agentic AI to help create PRD

Step #3: Generate the PRD

 + Project / PRD for PR Review Agent Creation 2934 of 3000 words used

Personal account

Start New Chat

Notifications 0

Chats 1

Documents 1

Projects New!

Templates

You have used 1 of 3 free chats.

Upgrade for \$5 / month


Upgrade Now

Recent All chats

Press **K** to search

Today

PRD for PR Review Agent Crea...

 **Ajay**
Free Trial




- Behavior: Can auto-approve, but transparency of actions/decisions is important
- Release: Start lean—minimum viable product

Here's how I suggest we structure the PRD for clarity and momentum. Does this match your expectations before I dive into the draft?


Outline Proposal:

- TL;DR:** Short summary of problem/solution
- Goals:** Business/user goals, non-goals (for focus)
- User Stories:** Developer workflow from both the dev building it and hypothetical external user
- Functional Requirements:** Key features—GitHub integration, code parsing, review feedback, auto-approval logic, transparency/logging
- User Experience:** End-to-end journey from PR creation to review to approval/notification
- Narrative:** Example scenario showing user value
- Success Metrics:** Adoption, speed, security, etc.
- Technical/Integration:** GitHub API, code analysis layers, etc.
- Milestones & Team:** Lean team/schedule/phases

Give me a thumbs up (or suggest tweaks) and I'll draft the full PRD in your template!




Please draft a full PRD



Finished writing MVP PR Review Agent for GitHub (Internal Release)




Edit





Your full PRD for the MVP PR Review Agent for GitHub is ready! It's structured around your priorities—internal developer users, architectural and security review, MVP scope, and a lean team. The document covers goals, user stories, requirements, UX, a narrative, metrics, technical considerations, and a realistic path to launch.

You can access and review the document in your doc switcher at the top of this chat (titled: "MVP PR Review Agent for GitHub (Internal Release)").


Want any tweaks, or ready to move on to outlining specific acceptance criteria or engineering tickets from this?



You are using the ChatPRD: PRD template





Send a message






Open in

Share

 1





Open in Editor

MVP PR Review Agent for GitHub (Internal Release)

TL;DR

Platformmetrics is building an MVP agent for internal use that automatically reviews GitHub pull requests for architectural alignment and security issues. The agent minimizes manual reviewer workload, accelerates delivery, and lays the groundwork for a productized solution for external stakeholders. Transparent, automated feedback improves PR quality from the start.

Goals

Business Goals

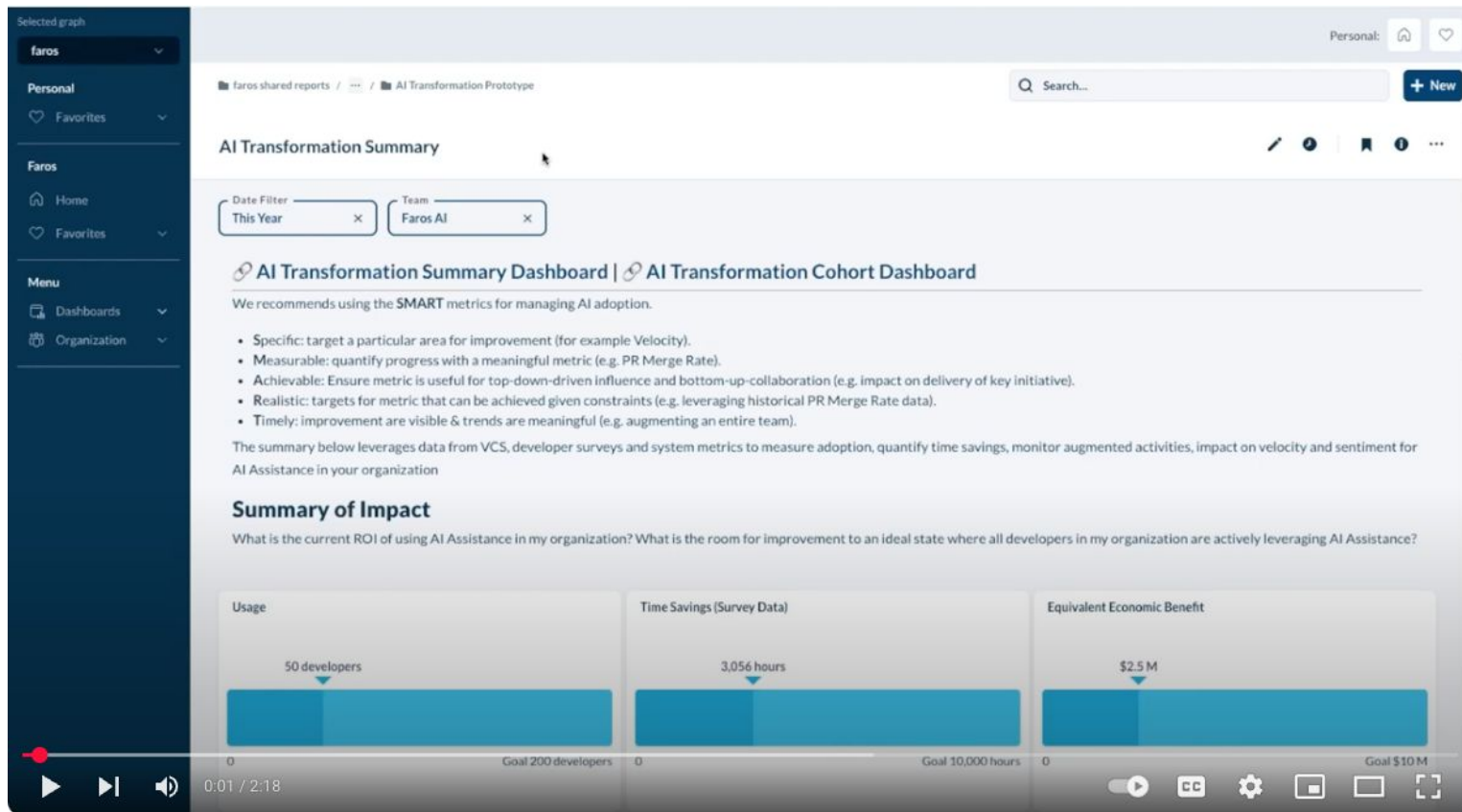
- Accelerate development cycles by reducing bottlenecks related to manual PR review.
- Improve overall code quality and security posture in the codebase.
- Validate the value and technical feasibility for possible external commercialization.
- Lay the foundation for showcasing "developer operations" automation capability to potential customers.
- Identify and document technical and process learnings to inform broader agent-based initiatives.

User Goals

- Receive prompt, actionable, and reliable feedback on architectural and security issues within PRs.
- Minimize friction and waiting time during PR review.
- Increase developer trust in code review automation by exposing agent decisions and reasoning.
- Maintain a clear and auditable record of agent actions for confidence and learning.
- Simplify PR approval flows (auto-approve with evident traceability for manual confirmation).

Next Goals

AI Transformation Summary Tools



Keys to adoption



Align to Developer Value Streams

- *Not random bots - align to build/deploy/troubleshoot/test flows*



Optimize for flow not features

- *Focus on seamlessness: IDE/Dev Portals embedded agents > dashboards*



Make agents collaborative not controlling

- *Agents propose, humans validate — human-in-the-loop.*



Observability and guardrails by default

- *LLM Observability, Trust Layers (Citations, RAG telemetry)*



Measure business impact, not model metrics

- *MTTR, Deployment Frequency, Dev Satisfaction – not just LLM accuracy.*

Effective Platform Engineering


- **Effective Platform Engineering** book by [Manning](#)
- MEAP currently out awaiting print version soon





More Information



<https://effectiveplatformengineering.com/>



Effective Platform Engineering





[Buy Now](#)


[Ajay Chankramath](#)



[Nic Cheneweth](#)



[Bryan Oliver](#)



[Sean Alvarez](#)


"Effective Platform Engineering" is a comprehensive guide that introduces platform engineering as a discipline, focusing on creating developer platforms that enhance team efficiency and streamline application deployment. The book provides practical insights into designing and managing platforms that bridge the gap between operations and development, automating tasks throughout the software development lifecycle. Readers will learn to build internal developer platforms and portals, ensuring seamless adoption and satisfaction among teams. The authors emphasize the importance of secure, scalable Kubernetes-based engineering platforms and offer strategies for implementing effective Service Level Objectives to boost trust and adoption. Additionally, the book explores cutting-edge integrations of Generative AI tools to enhance developer productivity, providing readers with the knowledge to leverage the latest advancements in code generation.

Through practical examples and real-world scenarios, "Effective Platform Engineering" demonstrates how platform engineering differs from traditional DevOps and the unique value it brings to organizations. The book delves into both patterns and anti-patterns of platform development, guiding readers in designing and deploying secure, scalable, and observable engineering platforms. With the inclusion of diagrams, code samples, and exercises, readers can visualize key concepts and solidify their understanding. This resource is tailored for DevOps engineers familiar with Kubernetes, cloud environments, and infrastructure-as-code, aiming to equip them with the skills to establish platforms that reduce workloads, improve consistency, and accelerate software delivery.

Discover how platform engineering is revolutionizing the developer experience and operational efficiency. [Learn more](#)